Aww yeah, Bootstrap 4 is coming!

JavaScript

Bring Bootstrap's components to life with over a dozen custom jQuery plugins. Easily include them all, or one by one.

Overview

Individual or compiled

Plugins can be included individually (using Bootstrap's individual `*.js` files), or all at once (using `bootstrap.js` or the minified `bootstrap.min.js`).

> ### Using the compiled JavaScript
> Both `bootstrap.js` and `bootstrap.min.js` contain all plugins in a single file. Include only one.

> ### Plugin dependencies
> Some plugins and CSS components depend on other plugins. If you include plugins individually, make sure to check for these dependencies in the docs. Also note that all plugins depend on jQuery (this means jQuery must be included **before** the plugin files). Consult our `bower.json` to see which versions of jQuery are supported.

Data attributes

You can use all Bootstrap plugins purely through the markup API without writing a single line of JavaScript. This is Bootstrap's first-class API and should be your first consideration when using a plugin.

That said, in some situations it may be desirable to turn this functionality off. Therefore, we also provide the ability to disable the data attribute API by unbinding all events on the document namespaced with `data-api`. This looks like this:

```
$(document).off('.data-api')
```

Alternatively, to target a specific plugin, just include the plugin's name as a namespace along with the data-api namespace like this:

```
$(document).off('.alert.data-api')
```

> ### Only one plugin per element via data attributes
> Don't use data attributes from multiple plugins on the same element. For example, a button cannot both have a tooltip and toggle a modal. To accomplish this, use a wrapping element.

Programmatic API

We also believe you should be able to use all Bootstrap plugins purely through the JavaScript API. All public APIs are single, chainable methods, and return the collection acted upon.

```
$('.btn.danger').button('toggle').addClass('fat')
```

All methods should accept an optional options object, a string which targets a particular method, or nothing (which initiates a plugin with default behavior):

```
$('#myModal').modal()                    // initialized with defaults
$('#myModal').modal({ keyboard: false })  // initialized with no keyboard
$('#myModal').modal('show')              // initializes and invokes show immediately
```

Each plugin also exposes its raw constructor on a `Constructor` property: `$.fn.popover.Constructor`. If you'd like to get a particular plugin instance, retrieve it directly from an element: `$('[rel="popover"]').data('popover')`.

Default settings

You can change the default settings for a plugin by modifying the plugin's `Constructor.DEFAULTS` object:

```
$.fn.modal.Constructor.DEFAULTS.keyboard = false // changes default for the modal plugin's `keyboard` option to false
```

No conflict

Sometimes it is necessary to use Bootstrap plugins with other UI frameworks. In these circumstances, namespace collisions can occasionally occur. If this happens, you may call `.noConflict` on the plugin you wish to revert the value of.

```
var bootstrapButton = $.fn.button.noConflict() // return $.fn.button to previously assigned value
$.fn.bootstrapBtn = bootstrapButton            // give $().bootstrapBtn the Bootstrap functionality
```

Events

Bootstrap provides custom events for most plugins' unique actions. Generally, these come in an infinitive and past participle form - where the infinitive (ex. `show`) is triggered at the start of an event, and its past participle form (ex. `shown`) is triggered on the completion of an action.

As of 3.0.0, all Bootstrap events are namespaced.

All infinitive events provide `preventDefault` functionality. This provides the ability to stop the execution of an action before it starts.

```
$('#myModal').on('show.bs.modal', function (e) {
  if (!data) return e.preventDefault() // stops modal from being shown
})
```

Version numbers

The version of each of Bootstrap's jQuery plugins can be accessed via the `VERSION` property of the plugin's constructor. For example, for the tooltip plugin:

```
$.fn.tooltip.Constructor.VERSION // => "3.3.7"
```

No special fallbacks when JavaScript is disabled

Bootstrap's plugins don't fall back particularly gracefully when JavaScript is disabled. If you care about the user experience in this case, use `<noscript>` to explain the situation (and how to re-enable JavaScript) to your users, and/or add your own custom fallbacks.

> ### Third-party libraries
>
> **Bootstrap does not officially support third-party JavaScript libraries** like Prototype or jQuery UI. Despite `.noConflict` and namespaced events, there may be compatibility problems that you need to fix on your own.

Transitions transition.js

About transitions

For simple transition effects, include `transition.js` once alongside the other JS files. If you're using the compiled (or minified) `bootstrap.js`, there is no need to include this—it's already there.

What's inside

Transition.js is a basic helper for `transitionEnd` events as well as a CSS transition emulator. It's used by the other plugins to check for CSS transition support and to catch hanging transitions.

Disabling transitions

Transitions can be globally disabled using the following JavaScript snippet, which must come after `transition.js` (or `bootstrap.js` or `bootstrap.min.js`, as the case may be) has loaded:

```
$.support.transition = false
```

Modals modal.js

Modals are streamlined, but flexible, dialog prompts with the minimum required functionality and smart defaults.

Multiple open modals not supported

Be sure not to open a modal while another is still visible. Showing more than one modal at a time requires custom code.

Modal markup placement

Always try to place a modal's HTML code in a top-level position in your document to avoid other components affecting the modal's appearance and/or functionality.

Mobile device caveats

There are some caveats regarding using modals on mobile devices. See our browser support docs for details.

Due to how HTML5 defines its semantics, the `autofocus` HTML attribute has no effect in Bootstrap modals. To achieve the same effect, use some custom JavaScript:

```
$('#myModal').on('shown.bs.modal', function () {
  $('#myInput').focus()
})
```

Examples

Static example

A rendered modal with header, body, and set of actions in the footer.

EXAMPLE


```
<div class="modal fade" tabindex="-1" role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>
        <h4 class="modal-title">Modal title</h4>
      </div>
      <div class="modal-body">
        <p>One fine body&hellip;</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div><!-- /.modal-content -->
  </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
```

Live demo

Toggle a modal via JavaScript by clicking the button below. It will slide down and fade in from the top of the page.

EXAMPLE

Launch demo modal

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>
        <h4 class="modal-title" id="myModalLabel">Modal title</h4>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

Make modals accessible

Be sure to add `role="dialog"` and `aria-labelledby="..."`, referencing the modal title, to `.modal`, and `role="document"` to the `.modal-dialog` itself.

Additionally, you may give a description of your modal dialog with `aria-describedby` on `.modal`.

Embedding YouTube videos

Embedding YouTube videos in modals requires additional JavaScript not in Bootstrap to automatically stop playback and more. See this helpful Stack Overflow post for more information.

Optional sizes

Modals have two optional sizes, available via modifier classes to be placed on a `.modal-dialog`.

EXAMPLE

Large modal Small modal

```
<!-- Large modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target=".bs-example-modal-lg">Large modal</button>

<div class="modal fade bs-example-modal-lg" tabindex="-1" role="dialog" aria-labelledby="myLargeModalLabel">
  <div class="modal-dialog modal-lg" role="document">
    <div class="modal-content">
      ...
    </div>
  </div>
</div>

<!-- Small modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target=".bs-example-modal-sm">Small modal</button>

<div class="modal fade bs-example-modal-sm" tabindex="-1" role="dialog" aria-labelledby="mySmallModalLabel">
  <div class="modal-dialog modal-sm" role="document">
    <div class="modal-content">
      ...
    </div>
  </div>
</div>
```

Remove animation

For modals that simply appear rather than fade in to view, remove the `.fade` class from your modal markup.

```
<div class="modal" tabindex="-1" role="dialog" aria-labelledby="...">
  ...
</div>
```

Using the grid system

To take advantage of the Bootstrap grid system within a modal, just nest `.row`s within the `.modal-body` and then use the normal grid system classes.

EXAMPLE

Launch demo modal

```
<div class="modal fade" tabindex="-1" role="dialog" aria-labelledby="gridSystemModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>
        <h4 class="modal-title" id="gridSystemModalLabel">Modal title</h4>
      </div>
      <div class="modal-body">
        <div class="row">
          <div class="col-md-4">.col-md-4</div>
          <div class="col-md-4 col-md-offset-4">.col-md-4 .col-md-offset-4</div>
        </div>
        <div class="row">
          <div class="col-md-3 col-md-offset-3">.col-md-3 .col-md-offset-3</div>
          <div class="col-md-2 col-md-offset-4">.col-md-2 .col-md-offset-4</div>
        </div>
        <div class="row">
          <div class="col-md-6 col-md-offset-3">.col-md-6 .col-md-offset-3</div>
        </div>
        <div class="row">
          <div class="col-sm-9">
            Level 1: .col-sm-9
            <div class="row">
              <div class="col-xs-8 col-sm-6">
                Level 2: .col-xs-8 .col-sm-6
              </div>
              <div class="col-xs-4 col-sm-6">
                Level 2: .col-xs-4 .col-sm-6
              </div>
            </div>
          </div>
        </div>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div><!-- /.modal-content -->
  </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
```

Varying modal content based on trigger button

Have a bunch of buttons that all trigger the same modal, just with slightly different contents? Use `event.relatedTarget` and HTML `data-*` attributes (possibly via jQuery) to vary the contents of the modal depending on which button was clicked. See the Modal Events docs for details on `relatedTarget`,

EXAMPLE

Open modal for @mdo | Open modal for @fat | Open modal for @getbootstrap ...more buttons...

```
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal" data-whatever="@mdo">Open modal for @mdo</button>
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal" data-whatever="@fat">Open modal for @fat</button>
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal" data-whatever="@getbootstrap">Open modal for
@getbootstrap</button>
...more buttons...

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>
        <h4 class="modal-title" id="exampleModalLabel">New message</h4>
      </div>
      <div class="modal-body">
        <form>
          <div class="form-group">
            <label for="recipient-name" class="control-label">Recipient:</label>
            <input type="text" class="form-control" id="recipient-name">
          </div>
          <div class="form-group">
            <label for="message-text" class="control-label">Message:</label>
            <textarea class="form-control" id="message-text"></textarea>
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Send message</button>
      </div>
    </div>
  </div>
</div>
```

```
$('#exampleModal').on('show.bs.modal', function (event) {
  var button = $(event.relatedTarget) // Button that triggered the modal
  var recipient = button.data('whatever') // Extract info from data-* attributes
  // If necessary, you could initiate an AJAX request here (and then do the updating in a callback).
  // Update the modal's content. We'll use jQuery here, but you could use a data binding library or other methods instead.
  var modal = $(this)
  modal.find('.modal-title').text('New message to ' + recipient)
  modal.find('.modal-body input').val(recipient)
})
```

Usage

The modal plugin toggles your hidden content on demand, via data attributes or JavaScript. It also adds `.modal-open` to the `<body>` to override default scrolling behavior and generates a `.modal-backdrop` to provide a click area for dismissing shown modals when clicking outside the modal.

Via data attributes

Activate a modal without writing JavaScript. Set `data-toggle="modal"` on a controller element, like a button, along with a `data-target="#foo"` or `href="#foo"` to target a specific modal to toggle.

```
<button type="button" data-toggle="modal" data-target="#myModal">Launch modal</button>
```

Via JavaScript

Call a modal with id `myModal` with a single line of JavaScript:

```
$('#myModal').modal(options)
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-backdrop=""`.

Name	type	default	description
backdrop	boolean or the string `'static'`	true	Includes a modal-backdrop element. Alternatively, specify `static` for a backdrop which doesn't close the modal on click.
keyboard	boolean	true	Closes the modal when escape key is pressed
show	boolean	true	Shows the modal when initialized.

Name	type	default	description
remote	path	false	**This option is deprecated since v3.3.0 and has been removed in v4.** We recommend instead using client-side templating or a data binding framework, or calling jQuery.load yourself. If a remote URL is provided, **content will be loaded one time** via jQuery's `load` method and injected into the `.modal-content` div. If you're using the data-api, you may alternatively use the `href` attribute to specify the remote source. An example of this is shown below: `<a data-toggle="modal" href="remote.html" data-target="#modal">Click me</a>`

Methods

`.modal(options)`

Activates your content as a modal. Accepts an optional options `object`.

```
$('#myModal').modal({
  keyboard: false
})
```

`.modal('toggle')`

Manually toggles a modal. **Returns to the caller before the modal has actually been shown or hidden** (i.e. before the `shown.bs.modal` or `hidden.bs.modal` event occurs).

```
$('#myModal').modal('toggle')
```

`.modal('show')`

Manually opens a modal. **Returns to the caller before the modal has actually been shown** (i.e. before the `shown.bs.modal` event occurs).

```
$('#myModal').modal('show')
```

`.modal('hide')`

Manually hides a modal. **Returns to the caller before the modal has actually been hidden** (i.e. before the `hidden.bs.modal` event occurs).

```
$('#myModal').modal('hide')
```

`.modal('handleUpdate')`

Readjusts the modal's positioning to counter a scrollbar in case one should appear, which would make the modal jump to the left.

Only needed when the height of the modal changes while it is open.

```
$('#myModal').modal('handleUpdate')
```

Events

Bootstrap's modal class exposes a few events for hooking into modal functionality.

All modal events are fired at the modal itself (i.e. at the `<div class="modal">`).

Event Type	Description
show.bs.modal	This event fires immediately when the `show` instance method is called. If caused by a click, the clicked element is available as the `relatedTarget` property of the event.
shown.bs.modal	This event is fired when the modal has been made visible to the user (will wait for CSS transitions to complete). If caused by a click, the clicked element is available as the `relatedTarget` property of the event.
hide.bs.modal	This event is fired immediately when the `hide` instance method has been called.
hidden.bs.modal	This event is fired when the modal has finished being hidden from the user (will wait for CSS transitions to complete).
loaded.bs.modal	This event is fired when the modal has loaded content using the `remote` option.

```
$('#myModal').on('hidden.bs.modal', function (e) {
  // do something...
})
```

Dropdowns dropdown.js

Examples

Add dropdown menus to nearly anything with this simple plugin, including the navbar, tabs, and pills.

Within a navbar

EXAMPLE

Within pills

EXAMPLE

Regular link Dropdown Dropdown Dropdown

Usage

Via data attributes or JavaScript, the dropdown plugin toggles hidden content (dropdown menus) by toggling the `.open` class on the parent list item.

On mobile devices, opening a dropdown adds a `.dropdown-backdrop` as a tap area for closing dropdown menus when tapping outside the menu, a requirement for proper iOS support. **This means that switching from an open dropdown menu to a different dropdown menu requires an extra tap on mobile.**

Note: The `data-toggle="dropdown"` attribute is relied on for closing dropdown menus at an application level, so it's a good idea to always use it.

Via data attributes

Add `data-toggle="dropdown"` to a link or button to toggle a dropdown.

```
<div class="dropdown">
  <button id="dLabel" type="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Dropdown trigger
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" aria-labelledby="dLabel">
    ...
  </ul>
</div>
```

To keep URLs intact with link buttons, use the `data-target` attribute instead of `href="#"`.

```
<div class="dropdown">
  <a id="dLabel" data-target="#" href="http://example.com" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">
    Dropdown trigger
    <span class="caret"></span>
  </a>

  <ul class="dropdown-menu" aria-labelledby="dLabel">
    ...
  </ul>
</div>
```

Via JavaScript

Call the dropdowns via JavaScript:

```
$('.dropdown-toggle').dropdown()
```

> **`data-toggle="dropdown"` still required**
>
> Regardless of whether you call your dropdown via JavaScript or instead use the data-api, `data-toggle="dropdown"` is always required to be present on the dropdown's trigger element.

Options

None

Methods

`$().dropdown('toggle')`

Toggles the dropdown menu of a given navbar or tabbed navigation.

Events

All dropdown events are fired at the `.dropdown-menu` 's parent element.

All dropdown events have a `relatedTarget` property, whose value is the toggling anchor element.

Event Type	Description
show.bs.dropdown	This event fires immediately when the show instance method is called.
shown.bs.dropdown	This event is fired when the dropdown has been made visible to the user (will wait for CSS transitions, to complete).
hide.bs.dropdown	This event is fired immediately when the hide instance method has been called.
hidden.bs.dropdown	This event is fired when the dropdown has finished being hidden from the user (will wait for CSS transitions, to complete).

```
$('#myDropdown').on('show.bs.dropdown', function () {
  // do something…
})
```

ScrollSpy scrollspy.js

Example in navbar

The ScrollSpy plugin is for automatically updating nav targets based on scroll position. Scroll the area below the navbar and watch the active class change. The dropdown sub items will be highlighted as well.

EXAMPLE

@fat

Ad leggings keytar, brunch id art party dolor labore. Pitchfork yr enim lo-fi before they sold out qui. Tumblr farm-to-table bicycle rights whatever. Anim keffiyeh carles cardigan. Velit seitan mcsweeney's photo booth 3 wolf moon irure. Cosby sweater lomo jean shorts, williamsburg hoodie minim qui you probably haven't heard of them et cardigan trust fund culpa biodiesel wes anderson aesthetic. Nihil tattooed accusamus, cred irony biodiesel keffiyeh artisan ullamco consequat.

@mdo

Veniam marfa mustache skateboard, adipisicing fugiat velit pitchfork beard. Freegan beard aliqua cupidatat mcsweeney's vero. Cupidatat four loko nisi, ea helvetica nulla carles. Tattooed cosby sweater food truck, mcsweeney's quis non freegan vinyl. Lo-fi wes anderson +1 sartorial. Carles non aesthetic exercitation quis gentrify. Brooklyn adipisicing craft beer vice keytar deserunt.

Usage

Requires Bootstrap nav

Scrollspy currently requires the use of a Bootstrap nav component for proper highlighting of active links.

Resolvable ID targets required

Navbar links must have resolvable id targets. For example, a `<a href="#home">home</a>` must correspond to something in the DOM like `<div id="home"></div>` .

Non- `:visible` target elements ignored

Target elements that are not `:visible` according to jQuery will be ignored and their corresponding nav items will never be highlighted.

Requires relative positioning

No matter the implementation method, scrollspy requires the use of `position: relative;` on the element you're spying on. In most cases this is the `<body>` . When scrollspying on elements other than the `<body>` , be sure to have a `height` set and `overflow-y: scroll;` applied.

Via data attributes

To easily add scrollspy behavior to your topbar navigation, add `data-spy="scroll"` to the element you want to spy on (most typically this would be the `<body>`). Then add the `data-target` attribute with the ID or class of the parent element of any Bootstrap `.nav` component.

```
body {
  position: relative;
}
```

```
<body data-spy="scroll" data-target="#navbar-example">
  ...
  <div id="navbar-example">
    <ul class="nav nav-tabs" role="tablist">
      ...
    </ul>
  </div>
  ...
</body>
```

Via JavaScript

After adding `position: relative;` in your CSS, call the scrollspy via JavaScript:

```
$('body').scrollspy({ target: '#navbar-example' })
```

Methods

`.scrollspy('refresh')`

When using scrollspy in conjunction with adding or removing of elements from the DOM, you'll need to call the refresh method like so:

```
$('[data-spy="scroll"]').each(function () {
  var $spy = $(this).scrollspy('refresh')
})
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-offset=""`.

Name	type	default	description
offset	number	10	Pixels to offset from top when calculating position of scroll.

Events

Event Type	Description
activate.bs.scrollspy	This event fires whenever a new item becomes activated by the scrollspy.

```
$('#myScrollspy').on('activate.bs.scrollspy', function () {
  // do something…
})
```

Togglable tabs tab.js

Example tabs

Add quick, dynamic tab functionality to transition through panes of local content, even via dropdown menus. **Nested tabs are not supported.**

EXAMPLE

Home Profile Dropdown

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica. Reprehenderit butcher retro keffiyeh dreamcatcher synth. Cosby sweater eu banh mi, qui irure terry richardson ex squid. Aliquip placeat salvia cillum iphone. Seitan aliquip quis cardigan american apparel, butcher voluptate nisi qui.

> **Extends tabbed navigation**
> This plugin extends the tabbed navigation component to add tabbable areas.

Usage

Enable tabbable tabs via JavaScript (each tab needs to be activated individually):

```
$('#myTabs a').click(function (e) {
  e.preventDefault()
  $(this).tab('show')
})
```

You can activate individual tabs in several ways:

```
$('#myTabs a[href="#profile"]').tab('show') // Select tab by name
$('#myTabs a:first').tab('show') // Select first tab
$('#myTabs a:last').tab('show') // Select last tab
$('#myTabs li:eq(2) a').tab('show') // Select third tab (0-indexed)
```

Markup

You can activate a tab or pill navigation without writing any JavaScript by simply specifying `data-toggle="tab"` or `data-toggle="pill"` on an element. Adding the `nav` and `nav-tabs` classes to the tab `ul` will apply the Bootstrap tab styling, while adding the `nav` and `nav-pills` classes will apply pill styling.

```
<div>

  <!-- Nav tabs -->
  <ul class="nav nav-tabs" role="tablist">
    <li role="presentation" class="active"><a href="#home" aria-controls="home" role="tab" data-toggle="tab">Home</a></li>
    <li role="presentation"><a href="#profile" aria-controls="profile" role="tab" data-toggle="tab">Profile</a></li>
    <li role="presentation"><a href="#messages" aria-controls="messages" role="tab" data-toggle="tab">Messages</a></li>
    <li role="presentation"><a href="#settings" aria-controls="settings" role="tab" data-toggle="tab">Settings</a></li>
  </ul>

  <!-- Tab panes -->
  <div class="tab-content">
    <div role="tabpanel" class="tab-pane active" id="home">...</div>
    <div role="tabpanel" class="tab-pane" id="profile">...</div>
    <div role="tabpanel" class="tab-pane" id="messages">...</div>
    <div role="tabpanel" class="tab-pane" id="settings">...</div>
  </div>

</div>
```

Fade effect

To make tabs fade in, add `.fade` to each `.tab-pane`. The first tab pane must also have `.in` to make the initial content visible.

```
<div class="tab-content">
  <div role="tabpanel" class="tab-pane fade in active" id="home">...</div>
  <div role="tabpanel" class="tab-pane fade" id="profile">...</div>
  <div role="tabpanel" class="tab-pane fade" id="messages">...</div>
  <div role="tabpanel" class="tab-pane fade" id="settings">...</div>
</div>
```

Methods

`$().tab`

Activates a tab element and content container. Tab should have either a `data-target` or an `href` targeting a container node in the DOM. In the above examples, the tabs are the `<a>`s with `data-toggle="tab"` attributes.

`.tab('show')`

Selects the given tab and shows its associated content. Any other tab that was previously selected becomes unselected and its associated content is hidden. **Returns to the caller before the tab pane has actually been shown** (i.e. before the `shown.bs.tab` event occurs).

```
$('#someTab').tab('show')
```

Events

When showing a new tab, the events fire in the following order:

1. `hide.bs.tab` (on the current active tab)
2. `show.bs.tab` (on the to-be-shown tab)
3. `hidden.bs.tab` (on the previous active tab, the same one as for the `hide.bs.tab` event)
4. `shown.bs.tab` (on the newly-active just-shown tab, the same one as for the `show.bs.tab` event)

If no tab was already active, then the `hide.bs.tab` and `hidden.bs.tab` events will not be fired.

Event Type	Description
show.bs.tab	This event fires on tab show, but before the new tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.
shown.bs.tab	This event fires on tab show after a tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.
hide.bs.tab	This event fires when a new tab is to be shown (and thus the previous active tab is to be hidden). Use `event.target` and `event.relatedTarget` to target the current active tab and the new soon-to-be-active tab, respectively.
hidden.bs.tab	This event fires after a new tab is shown (and thus the previous active tab is hidden). Use `event.target` and `event.relatedTarget` to target the previous active tab and the new active tab, respectively.

```
$('a[data-toggle="tab"]').on('shown.bs.tab', function (e) {
  e.target // newly activated tab
  e.relatedTarget // previous active tab
})
```

Tooltips tooltip.js

Inspired by the excellent jQuery.tipsy plugin written by Jason Frame; Tooltips are an updated version, which don't rely on images, use CSS3 for animations, and data-attributes for local title storage.

Tooltips with zero-length titles are never displayed.

Examples

Hover over the links below to see tooltips:

EXAMPLE

Tight pants next level keffiyeh you probably haven't heard of them. Photo booth beard raw denim letterpress vegan messenger bag stumptown. Farm-to-table seitan, mcsweeney's fixie sustainable quinoa 8-bit american apparel have a terry richardson vinyl chambray. Beard stumptown, cardigans banh mi lomo thundercats. Tofu biodiesel williamsburg marfa, four loko mcsweeney's cleanse vegan chambray. A really ironic artisan whatever keytar, scenester farm-to-table banksy Austin twitter handle freegan cred raw denim single-origin coffee viral.

Static tooltip

Four options are available: top, right, bottom, and left aligned.

EXAMPLE

Tooltip on the left Tooltip on the top Tooltip on the bottom Tooltip on the right

Four directions

EXAMPLE

Tooltip on left Tooltip on top Tooltip on bottom Tooltip on right

```
<button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="left" title="Tooltip on left">Tooltip on left</button>

<button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="top" title="Tooltip on top">Tooltip on top</button>

<button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="bottom" title="Tooltip on bottom">Tooltip on bottom</button>

<button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="right" title="Tooltip on right">Tooltip on right</button>
```

Opt-in functionality

For performance reasons, the Tooltip and Popover data-apis are opt-in, meaning **you must initialize them yourself**.

One way to initialize all tooltips on a page would be to select them by their `data-toggle` attribute:

```
$(function () {
  $('[data-toggle="tooltip"]').tooltip()
})
```

Usage

The tooltip plugin generates content and markup on demand, and by default places tooltips after their trigger element.

Trigger the tooltip via JavaScript:

```
$('#example').tooltip(options)
```

Markup

The required markup for a tooltip is only a `data` attribute and `title` on the HTML element you wish to have a tooltip. The generated markup of a tooltip is rather simple, though it does require a position (by default, set to `top` by the plugin).

```
<!-- HTML to write -->
<a href="#" data-toggle="tooltip" title="Some tooltip text!">Hover over me</a>

<!-- Generated markup by the plugin -->
<div class="tooltip top" role="tooltip">
  <div class="tooltip-arrow"></div>
  <div class="tooltip-inner">
    Some tooltip text!
  </div>
</div>
```

Multiple-line links

Sometimes you want to add a tooltip to a hyperlink that wraps multiple lines. The default behavior of the tooltip plugin is to center it horizontally and vertically. Add `white-space: nowrap;` to your anchors to avoid this.

Tooltips in button groups, input groups, and tables require special setting

When using tooltips on elements within a `.btn-group` or an `.input-group`, or on table-related elements (`<td>`, `<th>`, `<tr>`, `<thead>`, `<tbody>`, `<tfoot>`), you'll have to specify the option `container: 'body'` (documented below) to avoid unwanted side effects (such as the element growing wider and/or losing its rounded corners when the tooltip is triggered).

Don't try to show tooltips on hidden elements

Invoking `$(...).tooltip('show')` when the target element is `display: none;` will cause the tooltip to be incorrectly positioned.

Accessible tooltips for keyboard and assistive technology users

For users navigating with a keyboard, and in particular users of assistive technologies, you should only add tooltips to keyboard-focusable elements such as links, form controls, or any arbitrary element with a `tabindex="0"` attribute.

Tooltips on disabled elements require wrapper elements

To add a tooltip to a `disabled` or `.disabled` element, put the element inside of a `<div>` and apply the tooltip to that `<div>` instead.

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-animation=""`.

Name	Type	Default	Description
animation	boolean	true	Apply a CSS fade transition to the tooltip

Name	Type	Default	Description
container	string | false	false	Appends the tooltip to a specific element. Example: `container: 'body'`. This option is particularly useful in that it allows you to position the tooltip in the flow of the document near the triggering element - which will prevent the tooltip from floating away from the triggering element during a window resize.
delay	number | object	0	Delay showing and hiding the tooltip (ms) - does not apply to manual trigger type If a number is supplied, delay is applied to both hide/show Object structure is: `delay: { "show": 500, "hide": 100 }`
html	boolean	false	Insert HTML into the tooltip. If false, jQuery's `text` method will be used to insert content into the DOM. Use text if you're worried about XSS attacks.
placement	string | function	'top'	How to position the tooltip - top | bottom | left | right | auto. When "auto" is specified, it will dynamically reorient the tooltip. For example, if placement is "auto left", the tooltip will display to the left when possible, otherwise it will display right. When a function is used to determine the placement, it is called with the tooltip DOM node as its first argument and the triggering element DOM node as its second. The `this` context is set to the tooltip instance.
selector	string	false	If a selector is provided, tooltip objects will be delegated to the specified targets. In practice, this is used to enable dynamic HTML content to have tooltips added. See this and an informative example.
template	string	`'<div class="tooltip" role="tooltip"><div class="tooltip-arrow"></div><div class="tooltip-inner"></div></div>'`	Base HTML to use when creating the tooltip. The tooltip's `title` will be injected into the `.tooltip-inner`. `.tooltip-arrow` will become the tooltip's arrow. The outermost wrapper element should have the `.tooltip` class.
title	string | function	''	Default title value if `title` attribute isn't present. If a function is given, it will be called with its `this` reference set to the element that the tooltip is attached to.
trigger	string	'hover focus'	How tooltip is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space. `manual` cannot be combined with any other trigger.
viewport	string | object | function	{ selector: 'body', padding: 0 }	Keeps the tooltip within the bounds of this element. Example: `viewport: '#viewport'` or `{ "selector": "#viewport", "padding": 0 }` If a function is given, it is called with the triggering element DOM node as its only argument. The `this` context is set to the tooltip instance.

> **Data attributes for individual tooltips**
>
> Options for individual tooltips can alternatively be specified through the use of data attributes, as explained above.

Methods

`$().tooltip(options)`

Attaches a tooltip handler to an element collection.

`.tooltip('show')`

Reveals an element's tooltip. **Returns to the caller before the tooltip has actually been shown** (i.e. before the `shown.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip. Tooltips with zero-length titles are never displayed.

```
$('#element').tooltip('show')
```

`.tooltip('hide')`

Hides an element's tooltip. **Returns to the caller before the tooltip has actually been hidden** (i.e. before the `hidden.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip.

```
$('#element').tooltip('hide')
```

`.tooltip('toggle')`

Toggles an element's tooltip. **Returns to the caller before the tooltip has actually been shown or hidden** (i.e. before the `shown.bs.tooltip` or `hidden.bs.tooltip` event occurs). This is considered a "manual" triggering of the tooltip.

```
$('#element').tooltip('toggle')
```

`.tooltip('destroy')`

Hides and destroys an element's tooltip. Tooltips that use delegation (which are created using the `selector` option) cannot be individually destroyed on descendant trigger elements.

```
$('#element').tooltip('destroy')
```

Events

Event Type	Description
show.bs.tooltip	This event fires immediately when the `show` instance method is called.
shown.bs.tooltip	This event is fired when the tooltip has been made visible to the user (will wait for CSS transitions to complete).
hide.bs.tooltip	This event is fired immediately when the `hide` instance method has been called.
hidden.bs.tooltip	This event is fired when the tooltip has finished being hidden from the user (will wait for CSS transitions to complete).
inserted.bs.tooltip	This event is fired after the `show.bs.tooltip` event when the tooltip template has been added to the DOM.

```
$('#myTooltip').on('hidden.bs.tooltip', function () {
  // do something…
})
```

Popovers popover.js

Add small overlays of content, like those on the iPad, to any element for housing secondary information.

Popovers whose both title and content are zero-length are never displayed.

Plugin dependency

Popovers require the tooltip plugin to be included in your version of Bootstrap.

Opt-in functionality

For performance reasons, the Tooltip and Popover data-apis are opt-in, meaning **you must initialize them yourself**.

One way to initialize all popovers on a page would be to select them by their `data-toggle` attribute:

```
$(function () {
  $('[data-toggle="popover"]').popover()
})
```

Popovers in button groups, input groups, and tables require special setting

When using popovers on elements within a `.btn-group` or an `.input-group`, or on table-related elements (`<td>`, `<th>`, `<tr>`, `<thead>`, `<tbody>`, `<tfoot>`), you'll have to specify the option `container: 'body'` (documented below) to avoid unwanted side effects (such as the element growing wider and/or losing its rounded corners when the popover is triggered).

Don't try to show popovers on hidden elements

Invoking `$(...).popover('show')` when the target element is `display: none;` will cause the popover to be incorrectly positioned.

Popovers on disabled elements require wrapper elements

To add a popover to a `disabled` or `.disabled` element, put the element inside of a `<div>` and apply the popover to that `<div>` instead.

Multiple-line links

Sometimes you want to add a popover to a hyperlink that wraps multiple lines. The default behavior of the popover plugin is to center it horizontally and vertically. Add `white-space: nowrap;` to your anchors to avoid this.

Examples

Static popover

Four options are available: top, right, bottom, and left aligned.

EXAMPLE

Popover top

Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Popover right

Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Popover bottom

Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Popover left

Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Live demo

EXAMPLE

Click to toggle popover

```
<button type="button" class="btn btn-lg btn-danger" data-toggle="popover" title="Popover title" data-content="And here's some amazing content.
It's very engaging. Right?">Click to toggle popover</button>
```

Four directions

EXAMPLE

Popover on right | Popover on top | Popover on bottom | Popover on left

```
<button type="button" class="btn btn-default" data-container="body" data-toggle="popover" data-placement="left" data-content="Vivamus sagittis
lacus vel augue laoreet rutrum faucibus.">
  Popover on left
</button>

<button type="button" class="btn btn-default" data-container="body" data-toggle="popover" data-placement="top" data-content="Vivamus sagittis
lacus vel augue laoreet rutrum faucibus.">
  Popover on top
</button>

<button type="button" class="btn btn-default" data-container="body" data-toggle="popover" data-placement="bottom" data-content="Vivamus
sagittis lacus vel augue laoreet rutrum faucibus.">
  Popover on bottom
</button>

<button type="button" class="btn btn-default" data-container="body" data-toggle="popover" data-placement="right" data-content="Vivamus sagittis
lacus vel augue laoreet rutrum faucibus.">
  Popover on right
</button>
```

Dismiss on next click

Use the `focus` trigger to dismiss popovers on the next click that the user makes.

> **Specific markup required for dismiss-on-next-click**
>
> For proper cross-browser and cross-platform behavior, you must use the `<a>` tag, *not* the `<button>` tag, and you also must include the `role="button"` and `tabindex` attributes.

EXAMPLE

Dismissible popover

```
<a tabindex="0" class="btn btn-lg btn-danger" role="button" data-toggle="popover" data-trigger="focus" title="Dismissible popover" data-
content="And here's some amazing content. It's very engaging. Right?">Dismissible popover</a>
```

Usage

Enable popovers via JavaScript:

```
$('#example').popover(options)
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-animation=""`.

Name	Type	Default	Description
animation	boolean	true	Apply a CSS fade transition to the popover
container	string | false	false	Appends the popover to a specific element. Example: `container: 'body'`. This option is particularly useful in that it allows you to position the popover in the flow of the document near the triggering element - which will prevent the popover from floating away from the triggering element during a window resize.
content	string | function	''	Default content value if `data-content` attribute isn't present. If a function is given, it will be called with its `this` reference set to the element that the popover is attached to.
delay	number | object	0	Delay showing and hiding the popover (ms) - does not apply to manual trigger type If a number is supplied, delay is applied to both hide/show Object structure is: `delay: { "show": 500, "hide": 100 }`
html	boolean	false	Insert HTML into the popover. If false, jQuery's `text` method will be used to insert content into the DOM. Use text if you're worried about XSS attacks.

Name	Type	Default	Description
placement	string | function	'right'	How to position the popover - top | bottom | left | right | auto. When "auto" is specified, it will dynamically reorient the popover. For example, if placement is "auto left", the popover will display to the left when possible, otherwise it will display right. When a function is used to determine the placement, it is called with the popover DOM node as its first argument and the triggering element DOM node as its second. The `this` context is set to the popover instance.
selector	string	false	If a selector is provided, popover objects will be delegated to the specified targets. In practice, this is used to enable dynamic HTML content to have popovers added. See this and an informative example.
template	string	`'<div class="popover" role="tooltip"><div class="arrow"></div><h3 class="popover-title"></h3><div class="popover-content"></div></div>'`	Base HTML to use when creating the popover. The popover's `title` will be injected into the `.popover-title`. The popover's `content` will be injected into the `.popover-content`. `.arrow` will become the popover's arrow. The outermost wrapper element should have the `.popover` class.
title	string | function	''	Default title value if `title` attribute isn't present. If a function is given, it will be called with its `this` reference set to the element that the popover is attached to.
trigger	string	'click'	How popover is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space. `manual` cannot be combined with any other trigger.
viewport	string | object | function	{ selector: 'body', padding: 0 }	Keeps the popover within the bounds of this element. Example: `viewport: '#viewport'` or `{ "selector": "#viewport", "padding": 0 }` If a function is given, it is called with the triggering element DOM node as its only argument. The `this` context is set to the popover instance.

> ### Data attributes for individual popovers
> Options for individual popovers can alternatively be specified through the use of data attributes, as explained above.

Methods

`$().popover(options)`

Initializes popovers for an element collection.

`.popover('show')`

Reveals an element's popover. **Returns to the caller before the popover has actually been shown** (i.e. before the `shown.bs.popover` event occurs). This is considered a "manual" triggering of the popover. Popovers whose both title and content are zero-length are never displayed.

```
$('#element').popover('show')
```

`.popover('hide')`

Hides an element's popover. **Returns to the caller before the popover has actually been hidden** (i.e. before the `hidden.bs.popover` event occurs). This is considered a "manual" triggering of the popover.

```
$('#element').popover('hide')
```

`.popover('toggle')`

Toggles an element's popover. **Returns to the caller before the popover has actually been shown or hidden** (i.e. before the `shown.bs.popover` or `hidden.bs.popover` event occurs). This is considered a "manual" triggering of the popover.

```
$('#element').popover('toggle')
```

`.popover('destroy')`

Hides and destroys an element's popover. Popovers that use delegation (which are created using the `selector` option) cannot be individually destroyed on descendant trigger elements.

```
$('#element').popover('destroy')
```

Events

Event Type	Description
show.bs.popover	This event fires immediately when the `show` instance method is called.
shown.bs.popover	This event is fired when the popover has been made visible to the user (will wait for CSS transitions to complete).
hide.bs.popover	This event is fired immediately when the `hide` instance method has been called.
hidden.bs.popover	This event is fired when the popover has finished being hidden from the user (will wait for CSS transitions to complete).
inserted.bs.popover	This event is fired after the `show.bs.popover` event when the popover template has been added to the DOM.

```
$('#myPopover').on('hidden.bs.popover', function () {
  // do something…
})
```

Alert messages alert.js

Example alerts

Add dismiss functionality to all alert messages with this plugin.

When using a `.close` button, it must be the first child of the `.alert-dismissible` and no text content may come before it in the markup.

EXAMPLE

Holy guacamole! Best check yo self, you're not looking too good. ×

Oh snap! You got an error! ×

Change this and that and try again. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cras mattis consectetur purus sit amet fermentum.

Take this action | Or do this

Usage

Just add `data-dismiss="alert"` to your close button to automatically give an alert close functionality. Closing an alert removes it from the DOM.

```
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
  <span aria-hidden="true">&times;</span>
</button>
```

To have your alerts use animation when closing, make sure they have the `.fade` and `.in` classes already applied to them.

Methods

`$().alert()`

Makes an alert listen for click events on descendant elements which have the `data-dismiss="alert"` attribute. (Not necessary when using the data-api's auto-initialization.)

`$().alert('close')`

Closes an alert by removing it from the DOM. If the `.fade` and `.in` classes are present on the element, the alert will fade out before it is removed.

Events

Bootstrap's alert plugin exposes a few events for hooking into alert functionality.

Event Type	Description

close.bs.alert	This event fires immediately when the `close` instance method is called.
closed.bs.alert	This event is fired when the alert has been closed (will wait for CSS transitions to complete).

```
$('#myAlert').on('closed.bs.alert', function () {
  // do something…
})
```

Buttons button.js

Do more with buttons. Control button states or create groups of buttons for more components like toolbars.

Cross-browser compatibility

Firefox persists form control states (disabledness and checkedness) across page loads. A workaround for this is to use `autocomplete="off"` . See Mozilla bug #654072.

Stateful

Add `data-loading-text="Loading..."` to use a loading state on a button.

This feature is deprecated since v3.3.5 and has been removed in v4.

Use whichever state you like!

For the sake of this demonstration, we are using `data-loading-text` and `$().button('loading')` , but that's not the only state you can use. See more on this below in the `$().button(string)` documentation.

EXAMPLE

Loading state

```
<button type="button" id="myButton" data-loading-text="Loading..." class="btn btn-primary" autocomplete="off">
  Loading state
</button>

<script>
  $('#myButton').on('click', function () {
    var $btn = $(this).button('loading')
    // business logic...
    $btn.button('reset')
  })
</script>
```

Single toggle

Add `data-toggle="button"` to activate toggling on a single button.

Pre-toggled buttons need `.active` and `aria-pressed="true"`

For pre-toggled buttons, you must add the `.active` class and the `aria-pressed="true"` attribute to the `button` yourself.

EXAMPLE

Single toggle

```
<button type="button" class="btn btn-primary" data-toggle="button" aria-pressed="false" autocomplete="off">
  Single toggle
</button>
```

Checkbox / Radio

Add `data-toggle="buttons"` to a `.btn-group` containing checkbox or radio inputs to enable toggling in their respective styles.

> ### Preselected options need `.active`
> For preselected options, you must add the `.active` class to the input's `label` yourself.

> ### Visual checked state only updated on click
> If the checked state of a checkbox button is updated without firing a `click` event on the button (e.g. via `<input type="reset">` or via setting the `checked` property of the input), you will need to toggle the `.active` class on the input's `label` yourself.

EXAMPLE

Checkbox 1 (pre-checked) | Checkbox 2 | Checkbox 3

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-primary active">
    <input type="checkbox" autocomplete="off" checked> Checkbox 1 (pre-checked)
  </label>
  <label class="btn btn-primary">
    <input type="checkbox" autocomplete="off"> Checkbox 2
  </label>
  <label class="btn btn-primary">
    <input type="checkbox" autocomplete="off"> Checkbox 3
  </label>
</div>
```

EXAMPLE

Radio 1 (preselected) | Radio 2 | Radio 3

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-primary active">
    <input type="radio" name="options" id="option1" autocomplete="off" checked> Radio 1 (preselected)
  </label>
  <label class="btn btn-primary">
    <input type="radio" name="options" id="option2" autocomplete="off"> Radio 2
  </label>
  <label class="btn btn-primary">
    <input type="radio" name="options" id="option3" autocomplete="off"> Radio 3
  </label>
</div>
```

Methods

`$().button('toggle')`

Toggles push state. Gives the button the appearance that it has been activated.

`$().button('reset')`

Resets button state - swaps text to original text. **This method is asynchronous and returns before the resetting has actually completed.**

`$().button(string)`

Swaps text to any data defined text state.

```
<button type="button" id="myStateButton" data-complete-text="finished!" class="btn btn-primary" autocomplete="off">
  ...
</button>

<script>
  $('#myStateButton').on('click', function () {
    $(this).button('complete') // button text will be "finished!"
  })
</script>
```

Collapse collapse.js

Flexible plugin that utilizes a handful of classes for easy toggle behavior.

> **Plugin dependency**
> Collapse requires the transitions plugin to be included in your version of Bootstrap.

Example

Click the buttons below to show and hide another element via class changes:

- `.collapse` hides content
- `.collapsing` is applied during transitions
- `.collapse.in` shows content

You can use a link with the `href` attribute, or a button with the `data-target` attribute. In both cases, the `data-toggle="collapse"` is required.

EXAMPLE

Link with href | Button with data-target

```
<a class="btn btn-primary" role="button" data-toggle="collapse" href="#collapseExample" aria-expanded="false" aria-controls="collapseExample">
  Link with href
</a>
<button class="btn btn-primary" type="button" data-toggle="collapse" data-target="#collapseExample" aria-expanded="false" aria-
controls="collapseExample">
  Button with data-target
</button>
<div class="collapse" id="collapseExample">
  <div class="well">
    ...
  </div>
</div>
```

Accordion example

Extend the default collapse behavior to create an accordion with the panel component.

EXAMPLE

Collapsible Group Item #1

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

Collapsible Group Item #2

Collapsible Group Item #3

```
<div class="panel-group" id="accordion" role="tablist" aria-multiselectable="true">
  <div class="panel panel-default">
    <div class="panel-heading" role="tab" id="headingOne">
      <h4 class="panel-title">
        <a role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseOne" aria-expanded="true" aria-controls="collapseOne">
          Collapsible Group Item #1
        </a>
      </h4>
    </div>
    <div id="collapseOne" class="panel-collapse collapse in" role="tabpanel" aria-labelledby="headingOne">
      <div class="panel-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat
skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin
coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan
excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them
accusamus labore sustainable VHS.
      </div>
    </div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading" role="tab" id="headingTwo">
      <h4 class="panel-title">
        <a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTwo" aria-expanded="false" aria-
controls="collapseTwo">
          Collapsible Group Item #2
        </a>
      </h4>
    </div>
    <div id="collapseTwo" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwo">
      <div class="panel-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat
skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin
coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan
excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them
accusamus labore sustainable VHS.
      </div>
    </div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading" role="tab" id="headingThree">
      <h4 class="panel-title">
        <a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseThree" aria-expanded="false" aria-
controls="collapseThree">
          Collapsible Group Item #3
        </a>
      </h4>
    </div>
    <div id="collapseThree" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingThree">
      <div class="panel-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat
skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin
coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan
excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them
accusamus labore sustainable VHS.
      </div>
    </div>
  </div>
</div>
```

It's also possible to swap out `.panel-body` s with `.list-group` s.

Collapsible list group

Make expand/collapse controls accessible

Be sure to add `aria-expanded` to the control element. This attribute explicitly defines the current state of the collapsible element to screen readers and similar assistive technologies. If the collapsible element is closed by default, it should have a value of `aria-expanded="false"` . If you've set the collapsible element to be open by default using the `in` class, set `aria-expanded="true"` on the control instead. The plugin will automatically toggle this attribute based on whether or not the collapsible element has been opened or closed.

Additionally, if your control element is targeting a single collapsible element – i.e. the `data-target` attribute is pointing to an `id` selector – you may add an additional `aria-controls` attribute to the control element, containing the `id` of the collapsible element. Modern screen readers and similar assistive technologies make use of this attribute to provide users with additional shortcuts to navigate directly to the collapsible element itself.

Usage

The collapse plugin utilizes a few classes to handle the heavy lifting:

- `.collapse` hides the content
- `.collapse.in` shows the content
- `.collapsing` is added when the transition starts, and removed when it finishes

These classes can be found in `component-animations.less`.

Via data attributes

Just add `data-toggle="collapse"` and a `data-target` to the element to automatically assign control of a collapsible element. The `data-target` attribute accepts a CSS selector to apply the collapse to. Be sure to add the class `collapse` to the collapsible element. If you'd like it to default open, add the additional class `in`.

To add accordion-like group management to a collapsible control, add the data attribute `data-parent="#selector"`. Refer to the demo to see this in action.

Via JavaScript

Enable manually with:

```
$('.collapse').collapse()
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-parent=""`.

Name	type	default	description
parent	selector	false	If a selector is provided, then all collapsible elements under the specified parent will be closed when this collapsible item is shown. (similar to traditional accordion behavior - this is dependent on the `panel` class)
toggle	boolean	true	Toggles the collapsible element on invocation

Methods

`.collapse(options)`

Activates your content as a collapsible element. Accepts an optional options `object`.

```
$('#myCollapsible').collapse({
  toggle: false
})
```

`.collapse('toggle')`

Toggles a collapsible element to shown or hidden. **Returns to the caller before the collapsible element has actually been shown or hidden** (i.e. before the `shown.bs.collapse` or `hidden.bs.collapse` event occurs).

`.collapse('show')`

Shows a collapsible element. **Returns to the caller before the collapsible element has actually been shown** (i.e. before the `shown.bs.collapse` event occurs).

`.collapse('hide')`

Hides a collapsible element. **Returns to the caller before the collapsible element has actually been hidden** (i.e. before the `hidden.bs.collapse` event occurs).

Events

Bootstrap's collapse class exposes a few events for hooking into collapse functionality.

Event Type	Description
show.bs.collapse	This event fires immediately when the `show` instance method is called.
shown.bs.collapse	This event is fired when a collapse element has been made visible to the user (will wait for CSS transitions to complete).
hide.bs.collapse	This event is fired immediately when the `hide` method has been called.
hidden.bs.collapse	This event is fired when a collapse element has been hidden from the user (will wait for CSS transitions to complete).

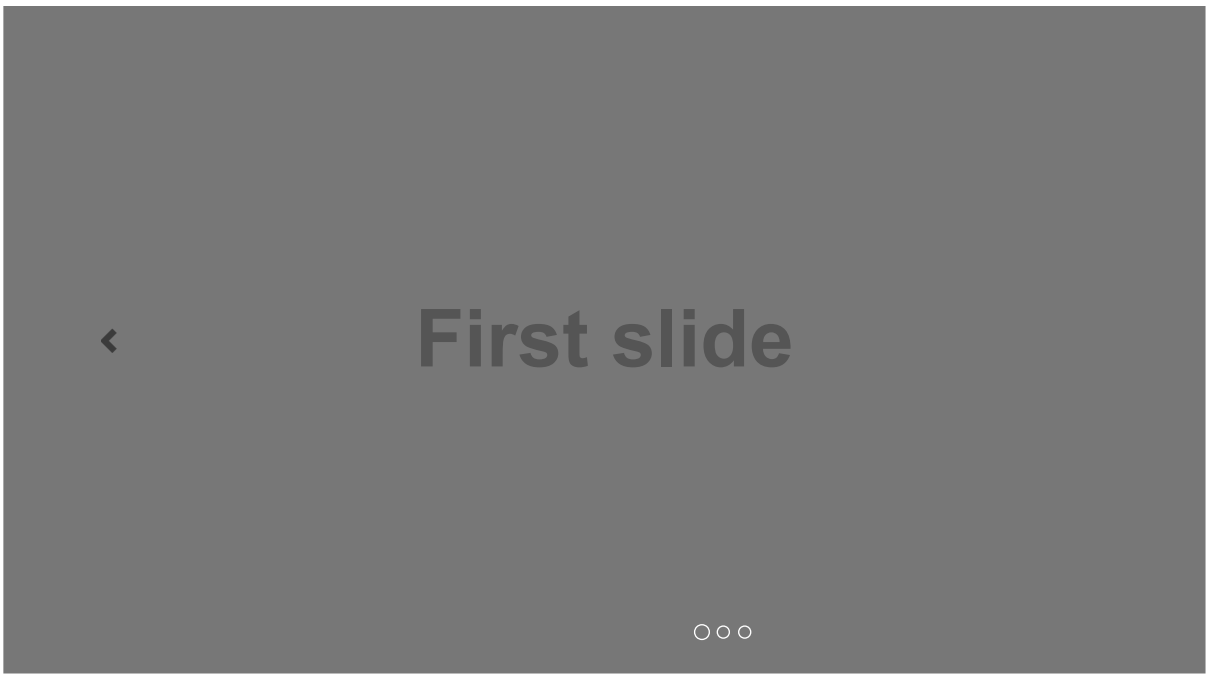
```
$('#myCollapsible').on('hidden.bs.collapse', function () {
  // do something…
})
```

Carousel carousel.js

A slideshow component for cycling through elements, like a carousel. **Nested carousels are not supported.**

Examples

EXAMPLE




```
<div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#carousel-example-generic" data-slide-to="0" class="active"></li>
    <li data-target="#carousel-example-generic" data-slide-to="1"></li>
    <li data-target="#carousel-example-generic" data-slide-to="2"></li>
  </ol>

  <!-- Wrapper for slides -->
  <div class="carousel-inner" role="listbox">
    <div class="item active">
      <img src="..." alt="...">
      <div class="carousel-caption">
        ...
      </div>
    </div>
    <div class="item">
      <img src="..." alt="...">
      <div class="carousel-caption">
        ...
      </div>
    </div>
    ...
  </div>

  <!-- Controls -->
  <a class="left carousel-control" href="#carousel-example-generic" role="button" data-slide="prev">
    <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="right carousel-control" href="#carousel-example-generic" role="button" data-slide="next">
    <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

Accessibility issue

The carousel component is generally not compliant with accessibility standards. If you need to be compliant, please consider other options for presenting your content.

Transition animations not supported in Internet Explorer 8 & 9

Bootstrap exclusively uses CSS3 for its animations, but Internet Explorer 8 & 9 don't support the necessary CSS properties. Thus, there are no slide transition animations when using these browsers. We have intentionally decided not to include jQuery-based fallbacks for the transitions.

Initial active element required

The `.active` class needs to be added to one of the slides. Otherwise, the carousel will not be visible.

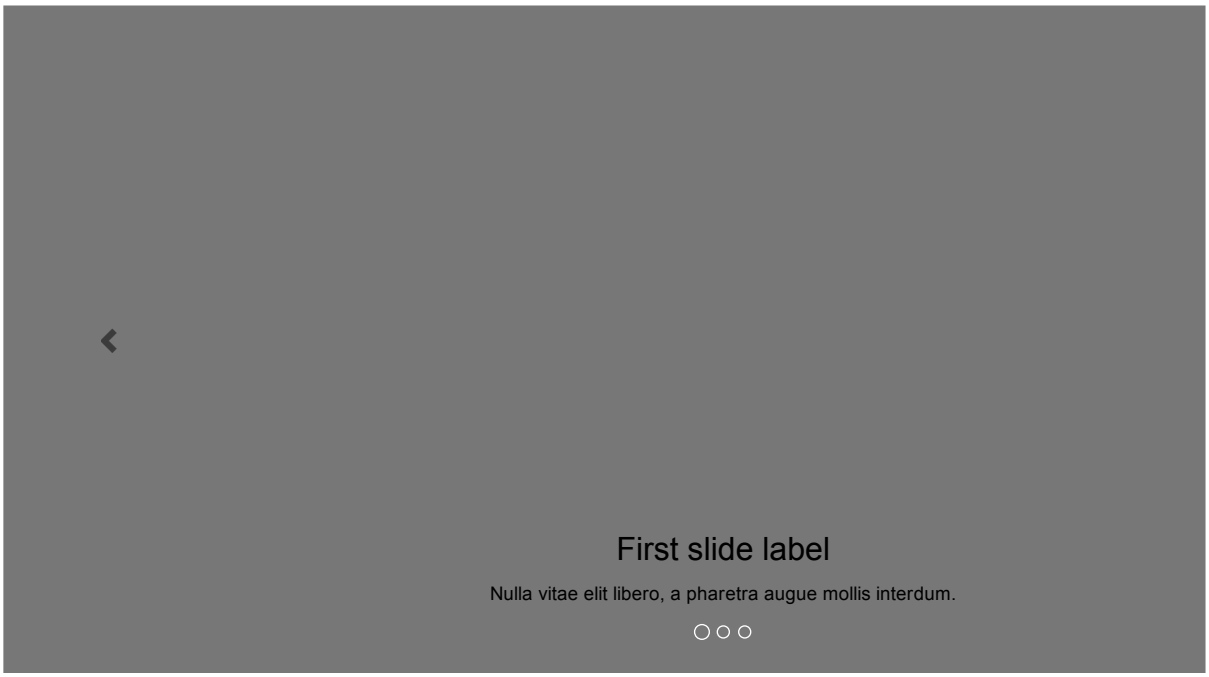
Glyphicon icons not necessary

The `.glyphicon .glyphicon-chevron-left` and `.glyphicon .glyphicon-chevron-right` classes are not necessarily needed for the controls. Bootstrap provides `.icon-prev` and `.icon-next` as plain unicode alternatives.

Optional captions

Add captions to your slides easily with the `.carousel-caption` element within any `.item`. Place just about any optional HTML within there and it will be automatically aligned and formatted.

EXAMPLE




```
<div class="item">
  <img src="..." alt="...">
  <div class="carousel-caption">
    <h3>...</h3>
    <p>...</p>
  </div>
</div>
```

Usage

Multiple carousels

Carousels require the use of an `id` on the outermost container (the `.carousel`) for carousel controls to function properly. When adding multiple carousels, or when changing a carousel's `id`, be sure to update the relevant controls.

Via data attributes

Use data attributes to easily control the position of the carousel. `data-slide` accepts the keywords `prev` or `next`, which alters the slide position relative to its current position. Alternatively, use `data-slide-to` to pass a raw slide index to the carousel `data-slide-to="2"`, which shifts the slide position to a particular index beginning with `0`.

The `data-ride="carousel"` attribute is used to mark a carousel as animating starting at page load. **It cannot be used in combination with (redundant and unnecessary) explicit JavaScript initialization of the same carousel.**

Via JavaScript

Call carousel manually with:

```
$('.carousel').carousel()
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-interval=""`.

Name	type	default	description
interval	number	5000	The amount of time to delay between automatically cycling an item. If false, carousel will not automatically cycle.
pause	string | null	"hover"	If set to `"hover"`, pauses the cycling of the carousel on `mouseenter` and resumes the cycling of the carousel on `mouseleave`. If set to `null`, hovering over the carousel won't pause it.
wrap	boolean	true	Whether the carousel should cycle continuously or have hard stops.
keyboard	boolean	true	Whether the carousel should react to keyboard events.

Methods

`.carousel(options)`

Initializes the carousel with an optional options `object` and starts cycling through items.

```
$('.carousel').carousel({
  interval: 2000
})
```

`.carousel('cycle')`

Cycles through the carousel items from left to right.

`.carousel('pause')`

Stops the carousel from cycling through items.

`.carousel(number)`

Cycles the carousel to a particular frame (0 based, similar to an array).

`.carousel('prev')`

Cycles to the previous item.

`.carousel('next')`

Cycles to the next item.

Events

Bootstrap's carousel class exposes two events for hooking into carousel functionality.

Both events have the following additional properties:

- `direction`: The direction in which the carousel is sliding (either `"left"` or `"right"`).
- `relatedTarget`: The DOM element that is being slid into place as the active item.

All carousel events are fired at the carousel itself (i.e. at the `<div class="carousel">`).

Event Type	Description
slide.bs.carousel	This event fires immediately when the `slide` instance method is invoked.
slid.bs.carousel	This event is fired when the carousel has completed its slide transition.

```
$('#myCarousel').on('slide.bs.carousel', function () {
  // do something…
})
```

Affix affix.js

Example

The affix plugin toggles `position: fixed;` on and off, emulating the effect found with `position: sticky;`. The subnavigation on the right is a live demo of the affix plugin.

Usage

Use the affix plugin via data attributes or manually with your own JavaScript. **In both situations, you must provide CSS for the positioning and width of your affixed content.**

Note: Do not use the affix plugin on an element contained in a relatively positioned element, such as a pulled or pushed column, due to a Safari rendering bug.

Positioning via CSS

The affix plugin toggles between three classes, each representing a particular state: `.affix`, `.affix-top`, and `.affix-bottom`. You must provide the styles, with the exception of `position: fixed;` on `.affix`, for these classes yourself (independent of this plugin) to handle the actual positions.

Here's how the affix plugin works:

1. To start, the plugin adds `.affix-top` to indicate the element is in its top-most position. At this point no CSS positioning is required.
2. Scrolling past the element you want affixed should trigger the actual affixing. This is where `.affix` replaces `.affix-top` and sets `position: fixed;` (provided by Bootstrap's CSS).
3. If a bottom offset is defined, scrolling past it should replace `.affix` with `.affix-bottom`. Since offsets are optional, setting one requires you to set the appropriate CSS. In this case, add `position: absolute;` when necessary. The plugin uses the data attribute or JavaScript option to determine where to position the element from there.

Follow the above steps to set your CSS for either of the usage options below.

Via data attributes

To easily add affix behavior to any element, just add `data-spy="affix"` to the element you want to spy on. Use offsets to define when to toggle the pinning of an element.

```
<div data-spy="affix" data-offset-top="60" data-offset-bottom="200">
  ...
</div>
```

Via JavaScript

Call the affix plugin via JavaScript:

```
$('#myAffix').affix({
  offset: {
    top: 100,
    bottom: function () {
      return (this.bottom = $('.footer').outerHeight(true))
    }
  }
})
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-offset-top="200"`.

Name	type	default	description
offset	number | function | object	10	Pixels to offset from screen when calculating position of scroll. If a single number is provided, the offset will be applied in both top and bottom directions. To provide a unique, bottom and top offset just provide an object `offset: { top: 10 }` or `offset: { top: 10, bottom: 5 }`. Use a function when you need to dynamically calculate an offset.
target	selector | node | jQuery element	the `window` object	Specifies the target element of the affix.

Methods

`.affix(options)`

Activates your content as affixed content. Accepts an optional options `object`.

```
$('#myAffix').affix({
  offset: 15
})
```

`.affix('checkPosition')`

Recalculates the state of the affix based on the dimensions, position, and scroll position of the relevant elements. The `.affix`, `.affix-top`, and `.affix-bottom` classes are added to or removed from the affixed content according to the new state. This method needs to be called whenever the dimensions of the affixed content or the target element are changed, to ensure correct positioning of the affixed content.

```
$('#myAffix').affix('checkPosition')
```

Events

Bootstrap's affix plugin exposes a few events for hooking into affix functionality.

Event Type	Description
affix.bs.affix	This event fires immediately before the element has been affixed.
affixed.bs.affix	This event is fired after the element has been affixed.
affix-top.bs.affix	This event fires immediately before the element has been affixed-top.
affixed-top.bs.affix	This event is fired after the element has been affixed-top.
affix-bottom.bs.affix	This event fires immediately before the element has been affixed-bottom.
affixed-bottom.bs.affix	This event is fired after the element has been affixed-bottom.